REMARKS

These remarks are in response to the Office Action dated July 15, 2004. Claims 1-15 are pending in the present Application. Claims 1-15 have been rejected. Claims 1-15 remain pending. For the reasons set forth more fully below, Applicant respectfully submits that the claims as presented are allowable. Consequently, reconsideration, allowance, and passage to issue are respectfully requested.

Double Patenting - Terminal Disclaimer

The Examiner has stated:

> **3. Claims 5, 10, 15 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 6, and 18 of copending Application No. 09/903937 (hereinafter '937). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations.**
>
> **Following is some examples of conflicting claims:**
>
> **As per instant claim 5, copending '937 claim 6 also recites a method including a plurality of directories, such method for compiling a file in the directories, and providing a master array of dependencies of the directories (e.g., merging the dependency array with the master array of changes); providing a code change to provide an updated program (e.g. providing a directory update mechanism); providing associated dependency changes to the master array; and steps for**
>
> **providing (dependency changes),**
> **merging (with master array);**
> **obtaining (dependency change);**
> **determining (change is in a directory)**
> **updating (directory in the master array)**
> **adding (dependency change)**
> **determining (another dependency change)**
> **repeating (steps... until all dependency..).**
>
> **But '937 claim does not recite compiling the updated program utilizing the updated master array wherein the directories file are compiled in an ordered manner based upon the dependencies of the pluralities of directories.**
>
> **However, '937 claim 6 recites an update mechanism for assigning the directories wherein the associated dependencies are provided and subsequent update changes merged into the master array and assigning a directory to a next available processor in an ordered manner to allow the processor to compile one file in the directory; hence suggests a program operable with a plurality of directories (i.e. update mechanism program) being adjusted to enable file in a directory to be compiled in an ordered manner according to the dependencies changes provided to the program. Hence, it would have been obvious for one skill**

in the art at the time the invention was made to provide the update mechanism as a compilable software program, and compiling of such updated program (upon dependencies changes being made thereto) so as to utilize the updated dependencies (in the master array) for enabling an ordered manner of file compilation based thereupon.

As for instant clam 10, '937 claim 18 recites the same limitations and obvious variations of claim 10 (e.g. compiling an updated program); all of these features having been addressed above.

As per instant claim 15 (a computer-readable medium version of claim 10), '937 claim 18 recites the same limitations and obvious variations of claim 10 as addressed above.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

In response, Applicant will file a terminal disclaimer upon determinability of allowance of the claims in view of the prior art.

Claim Rejections - 35 U.S.C. §102

The Examiner has stated:

5. Claims 1, 3-4, 6, 8-9, 11, and 13-14 are rejected under 35 U.S.C. 102(b) as being anticipated by Kionka, USPN: 5,361,357 (hereinafter Kionka).

As per claim 1, Kionka discloses a method for minimizing the cycle time when compiling a program in a computer system (e.g., optimizing, efficient compilation – col. 1, lines 7-25), the program including a plurality of directories (e.g. Fig. 2a) and each of the directories including a code file; the method comprising:

providing a master array of directories of the program (e.g. abstracted tree register 48- col. 6, lines 21-68; Fig. 2C), wherein the master array lists the dependencies of the directories, providing a code change to the program to provide an updated program (e.g. super Makefile; modified smf-col. 7-8; groups of Makefiles out-of-date, doOneMakefile- col. 11-12- Note: compounding individual Make into an updated one Make file is updating with code change;)

providing associated dependency changes to the master array to provide an updated master array (e.g. are to be updated-col. 6, lines 60-68); and

But Kionka does not explicitly disclose compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories. However, Kionka discloses a equivalent of a master Makefile integrating all the individual directory Makefiles (e.g. col. 6, lines 31-41; Appendix – col. 7-26); hence has implicitly disclosed compiling of an updated and master Makefile using the order of dependencies as adjusted from integrating the plurality of directories as set forth in the individual Makefiles. Therefore, the limitation is disclosed.

As per claim 3, Kionka discloses that the associated dependencies changes are provided via a directory update mechanism (e.g....are to be updated – col. 6, lines 42-68; col. 7-8; col. 11-12).

As per claim 4, Kionka discloses the steps of:
providing an array of dependency changes (e.g. Directory Description register 50 – Fig. 2a-c; registers 40, 46, 56 – Fig. 1; col. 5, lines 60-67); and
merging the dependency changes array with a master array of changes (e.g. Abstract Tree Register 48, Output file Register 54-Fig. 1).
As per claim 6, Kionka discloses a system for minimizing the cycle time when compiling a program in a computer system, the program including a plurality of directories and each of the directories including a code file, the method comprises the steps of
providing (a master array),
providing (an updated program);
providing (dependencies changes...updated master array); and
compiling (the updated program... an ordered manner); all of which steps having been addressed in claim 1.
As per claims 8-9, these claims correspond to claims 3-4; hence are rejected using the same rejection as set forth therein, respectively.
As per claim 11, this is the computer-readable medium version of method claim 1, including the same step limitations; hence is rejected using the corresponding rejection as set forth therein.
As per claims 13-14, these claims correspond to claims 3-4; hence are rejected using the same rejection as set forth therein, respectively.

Applicant respectfully disagrees with the Examiner's rejections. The present invention provides a method and system for minimizing the cycle time when compiling a program in a computer system, where a program includes a plurality of directories and each of the directories includes a code file. In accordance with the present invention, the method includes: (a) providing a master array of directories of the program, wherein the master array lists the dependencies of the directories; (b) providing a code change to the program to provide an updated program; (c) providing associated dependency changes to the master array to provide an updated master array; and (d) compiling the updated program utilizing the updated master array wherein the code files of the directories are compiled in an ordered manner based upon the dependencies of the plurality of directories. As a result, compile cycle time for large programs is significantly reduced. A second advantage is that since the dependencies are updated substantially simultaneously with code changes, there are minimal dependency violations and therefore few deadlocks. Kionka does not teach or suggest these features, as discussed below.

Kionka discloses a method and apparatus for optimizing the sequencing and time requirements for compiling large sets of source code residing in multiple hierarchical file directories using an abstracted logical description of the hierarchical file relations existing between directories. The system consists of a logic processor working in concert with input and output file registers, a match register, and an abstracted tree register for the purpose of creating a identifying, comparing, and sequencing file names in a final description of the global directory. The method iteratively identifies the primary input files and the intermediate input files for a given output file for each of a series of directories, inverts the casual relationship between the output file and its intermediary input files, and accumulates and stores these relationships in a sequential manner for subsequent use (Summary).

Applicant respectfully submits that Kionka does not teach the combination of "providing a master array of directories of the program, wherein the master array lists the dependencies of the directories," and "providing associated dependency changes to the master array to provide an updated master array," as recited in independent claims 1, 6, and 11. The Examiner has referred to updates in column 6, lines 60-68, of Kionka. However, the updates of Kionka are not updates to directories as in the present invention but are instead updates to files. Kionka explicitly teaches that the "various files in the different directories are to be updated" (column 6, lines 64-66). In other words, while the files of Kionka are in directories, the files are not the same as directories. Accordingly, the file dependencies of Kionka are different from the directory dependencies of the present invention.

Applicant agrees with the Examiner that Kionka does not explicitly disclose "compiling the updated program utilizing the updated master array wherein the code files of the directories are

compiled in an ordered manner based upon the dependencies of the plurality of directories," as recited in independent claims 1, 6, and 11. The Examiner has asserted that this step is implicitly disclosed and has referred to a "master Makefile." However, Kionka does not describe a "master" Makefile, and the term "makefile" is not directed to making directories but is instead directed to making files. Kionka describes a rule that defines a "causal relationship between the files," where the "rule for the files in an abstracted tree will often by "make [filename]" (column 6, lines 31-41). Kionka further states that the abstracted tree is used "as input file describing the sequence by which the various files in the different directories are to be updated (column 6, lines 60-66). Accordingly, any compiling in Kionka is not based upon dependencies of directories as in the present invention but is instead based upon dependencies of files.

Therefore, Kionka does not teach or suggest the combination of steps as recited in independent claims 1, 6, and 11, and these claims are allowable over Kionka.

Claim Rejections - 35 U.S.C. §103

The Examiner has stated:

7. Claims 2, 7, 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kionka, USPN: 5,361,357 (hereinafter Kionka), as applied to claims 1, 6, 11; in view of Hanna et al., USPN: 5,748,961 (hereinafter Hanna), and further in view of Chase, Jr., et al., USPN: 4,951,192 (hereinafter Chase).

As per claim 2, Kionka does not explicitly teach a scheduler being utilized to compile the update program, wherein the scheduler receives the dependency changes and a list of processors from a processor array. Kionka, however, mentions about complexity of a system with a larger amount of dependencies and LAN connectivity with more than one processors to distribute storage burden (e.g., Fig. 1). Hanna, in a method to optimize compiling and expedite the object code building in a large system, using a Makefile (ch. 9.2-col. 41) in conjunction with build models or a tree structure of directory models (e.g. Fig. 1a-b) analogous to a master array of directories, teaches a schedule process as to how to distribute the code building process (ch. 10.6-col. 45). The distribution of resources and tasks in a multi-computing network or complex interconnected system using a resource scheduler as suggested by Hanna is further enhanced via the teachings by Chase to provide parallel compilation. Chase, in a system to configure a large software

system analogous to Kionka or Hanna, discloses a scheduler to choose and assign available or most suitable processors from a displayed list to compile the buildable components (e.g. col. 1, line 67 to col. 2, line 25). It would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance the complex system connecting a plural computer for handling resources as suggested by Kionka and Hanna so that a scheduling process as taught by Hanna can further be used in compiling task assignment via selection from a plurality of processors being listed as taught by Chase because that way more resources can be distributed and tasks can be imparted separately alleviating thereby the condition of dependency between subsystem task that might otherwise be a limiting factor to expediting the building process and product delivery of a large software system (see Chase: impairs productivity, independent components... parallel – col. 1-2).

As per claims 7 and 12, these are claims corresponding to claim 2, and are rejected using the same rationale as set forth therein.

8. Claims 5, 10 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kionka, USPN: 5,361,357, as applied to claims 4, 9, 14; in view of Hanna et al., USPN: 5,748,961.

As per claim 5, Kionka discloses the steps of:

obtaining a dependency change from the dependency changes array (e.g. step 29-Fig. 2a-c; Input: %rels dependencies for each target – col. 19-20; 21-22 – Note: code change in Makefile is equivalent to obtaining a dependency change, a set of dependent elements being sequences into a list, when processing the directories);

determining whether the dependency change is in a directory in the master array (e.g. exists, die (Not implemented), $fake_top_name not needed-col. 21-22- Note: making adjustment in the master Makefile is equivalent to implementing changes being reflected in the directory element comprising the master array);

(i) updating the directory in the master array of the dependency change in a directory of the master array (e.g. col. 6, lines 60-67);

(iii) determining if there is another dependency change in the dependency changes array after step (i) (see Fig. 2a-c; Appendix code with foreach); and

repeating steps (i) and (iii) until all dependency changes have been obtained from the dependency change array.

But Kionka does not explicitly disclose the step of (ii) adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array; nor does Kionka disclose repeating of steps (i), (ii) and (iii).

However, Kionka discloses that the master array includes each directory files relationship as those directories are sequentially inputted for processing (e.g. Fig. 2a-c and related text) and teaches a the list of added dependent elements pertinent to the file directory structures being inputted to form the final master structure (e.g. @new_list = shift (); push (@new_list, $top_target – col. 21-22; Fig. 2a, step 31); hence has suggested the idea of creating a list of elements coming from a specific directory and of adding a list dependency changes to the master array if the change in the array/list of dependency change has not yet been included in the master array (Note: list of dependency element, e.g. $rels Input: %rels dependencies for each target-col. 21-22, is equivalent to array being hierarchized according to priority and rules of a Makefile). Thus, the concept of adding a array of dependency elements or directory-related elements (re Fig. 2a, step 31) into a grouping or any hierarchy of elements being stored and arrayed is

suggested, i.e., a hierarchy of dependency being grouped and arrayed by its directory root. Further, Hanna, in the system as mentioned in claim 1, supports grouping of elements into the models in terms of directories (e.g. Fig. 1b).

Based on the teachings by Hanna and on the suggested concepts and master array upgrade teachings by Kionka from above, it would have been obvious for one skill in the art at the time the invention was made to provide the creation of a new change directory within the master array as suggested by Hanna or Kionka so as to include successive dependency changes or array of dependency elements into such form of directory grouping as suggested above and provide the repeating of steps from (i) to (iii) because generating a set of dependent elements being added to a directory, a separate grouping structure or a flattened hierarchy of elements would provide a better view to the abstract tree, i.e. the global master array, since this master array is purposed for enabling a facilitated perception of a more complex system wherein dependency of directories being integrated in the final build is subjected to continual and dynamic updates or adjustments.

As per claims 10 and 15, these are claims corresponding to claim 5, and are rejected using the same rationale as set forth therein.

Applicant respectfully disagrees with the Examiner's rejections. Dependent claims 2, 5, 7, 10, 12, and 15 depend from independent claims 1, 6, and 11, respectively. Accordingly, the above-articulated arguments related to independent claims 1, 6, and 11 apply with equal force to claims 2, 5, 7, 10, 12, and 15, which are thus allowable over the cited reference for at least the same reasons as claims 1, 6, and 11.

Remaining dependent claims

Dependent claims 3-4, 8-9, and 13-14 depend from independent claims 1, 6, and 11, respectively. Accordingly, the above-articulated arguments related to independent claims 1, 6, and 11 apply with equal force to claims 3-4, 8-9, and 13-14, which are thus allowable over the cited references for at least the same reasons as claims 1, 6, and 11.

Conclusion

In view of the foregoing, Applicant submits that claims 1-15 are patentable over the cited references. Applicant, therefore, respectfully requests reconsideration and allowance of the claims as now presented.

Applicant's attorney believes that this application is in condition for allowance. Should any unresolved issues remain, the Examiner is invited to call Applicant's attorney at the telephone number indicated below.

Respectfully submitted,

SAWYER LAW GROUP LLP

October 15, 2004
Date

Joseph A. Sawyer, Jr.
Attorney for Applicant(s)
Reg. No. 30,801
(650) 493-4540